RETURN ZER0: APLIKASI PERMAINAN PENGASAH LOGIKA PEMROGRAMAN BERBASIS CLI (COMMAND LINE INTERFACE)

LAPORAN

Disusun untuk memenuhi UAS mata kuliah

Konsep Pemrograman (12013120406)

Dosen Pengampu:

Prof. Drs. Bambang Harjito, M.App.Sc., Ph.D.

Disusun oleh:

Kelompok 01

1. Gamma Assyafi Fadhillah Ar Rasyad (L0125013)
2. Jaffan Arya Wirasena (L0125017)
3. Linda Sihmawati (L0125049)

KELAS A

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET SURAKARTA

202

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Deskripsi Sistem

Return Zer0 adalah sebuah perangkat lunak permainan edukasi berbasis konsol (*console-based application*) yang dikembangkan menggunakan bahasa pemrograman C. Sistem ini dirancang sebagai media evaluasi interaktif untuk menguji pemahaman pengguna terhadap konsep dasar pemrograman, khususnya sintaks dan logika dalam bahasa C.

Sistem permainan ini mengadopsi struktur kuis pilihan ganda dengan tingkat kesulitan berjenjang (progresif), dimulai dari level Dasar, Menengah, hingga Mahir. Keunikan dari sistem ini terletak pada fitur Level Bonus C++ yang bersifat kondisional; hanya dapat diakses apabila pemain memenuhi ambang batas jawaban benar tertentu (lebih dari 12 jawaban benar).

Secara teknis, sistem ini menerapkan konsep manipulasi file (*file handling*) yang ekstensif. Data soal tidak ditanam di dalam kode program (*hardcoded*), melainkan dibaca dari file eksternal (soal.txt), memungkinkan pembaruan konten tanpa perlu kompilasi ulang. Selain itu, sistem dilengkapi dengan fitur persistensi data yang memungkinkan pemain menyimpan progres permainan (*Save Game*) ke dalam file save.txt dan melanjutkan kembali di sesi berikutnya, serta sistem Papan Peringkat (*Leaderboard*) yang mencatat 5 skor tertinggi menggunakan algoritma pengurutan *Bubble Sort*.

1.2 Library yang Digunakan

Untuk mendukung fungsionalitas program, Return Zer0 memanfaatkan beberapa pustaka (*library*) standar dan spesifik sistem operasi sebagai berikut:

1. <stdio.h> (Standard Input Output)

   Digunakan sebagai fondasi utama interaksi program, mencakup fungsi printf untuk menampilkan antarmuka dan soal ke layar, scanf untuk menerima input jawaban pengguna, serta fungsi manajemen file (fopen, fclose, fgets, fprintf) untuk membaca *database* soal dan menyimpan progres pemain.

2. <stdlib.h> (Standard Library)

   Digunakan untuk manajemen memori dan utilitas sistem. Fungsi penting yang digunakan meliputi system("cls") atau system("clear") untuk membersihkan

layar terminal agar antarmuka tetap rapi, atoi untuk mengonversi *string* menjadi *integer* saat memproses data dari file, dan exit untuk terminasi program.

3. <string.h>

   Sangat krusial dalam program ini untuk memanipulasi data teks. Fungsi utamanya meliputi strcpy untuk menyalin data nama dan soal ke dalam variabel *struct*, strcmp untuk perbandingan, strlen untuk validasi panjang nama pemain, dan strtok untuk memecah (*parsing*) baris data dari file teks (format: Pertanyaan#Opsi#Jawaban) menjadi token-token terpisah.

4. <windows.h> (Windows) atau <unistd.h> (Linux/Unix)

   Digunakan untuk memberikan efek jeda waktu (*delay*) pada transisi antar layar menggunakan fungsi Sleep() atau usleep(), memberikan pengalaman pengguna (*User Experience*) yang lebih halus dan tidak terburu-buru.

1.3 Alur Kerja Program

Alur kerja sistem Return Zer0 dapat dideskripsikan melalui tahapan proses berikut:

1. Inisialisasi & Validasi Data

   Saat program dijalankan, sistem akan mencoba memuat file eksternal (soal.txt). Jika file tidak ditemukan pada direktori yang ditentukan (../test/), program akan memberikan pesan kesalahan dan berhenti. Jika berhasil, data soal akan dimuat ke dalam memori (*array of struct*).

2. Menu Utama

   Pengguna disajikan menu utama yang berisi opsi: Mulai Baru, Lanjutkan, Leaderboard, Credits, dan Keluar.

3. Proses Permainan (Game Loop)
   - Input Data: Pemain memasukkan nama (validasi maks. 15 karakter).
   - Looping Soal: Program menampilkan soal satu per satu sesuai urutan array.
   - Mekanisme Jawab: Pemain menginput jawaban (1-4). Jika benar, skor bertambah sesuai bobot level. Jika salah, nyawa (HP) berkurang.
   - Kondisi Kalah: Jika nyawa mencapai 0, permainan berakhir (Game Over) dan data save otomatis dihapus.
   - Simpan & Keluar: Di tengah permainan, pemain dapat memilih opsi simpan (input '9') untuk menyimpan status level, skor, dan nyawa ke file save.txt.

4. Sistem Leveling & Bonus

Sistem secara otomatis mendeteksi indeks soal untuk menentukan level (Dasar/Menengah/Mahir). Setelah soal ke-15, sistem mengecek jumlah jawaban benar. Jika >12, Level Bonus C++ terbuka; jika tidak, permainan selesai.

5. Penyelesaian & Perekaman Skor

Setelah permainan selesai atau Game Over, sistem membandingkan skor akhir pemain dengan data di leaderboard.txt. Jika skor masuk dalam 5 besar, daftar akan diperbarui, diurutkan ulang, dan disimpan kembali.

BAB II
PENJELASAN SOURCE CODE

Pengembangan aplikasi permainan "Return Zer0" ini menggunakan kode program utama yang disimpan dalam file main.c. File ini mencakup seluruh logika permainan, mulai dari manajemen memori, struktur data, antarmuka pengguna berbasis teks (CLI), hingga operasi file handling untuk penyimpanan data. Berikut adalah rincian fungsionalitas di dalam main.c:

1. Library dan Preprocessor Directives
2. Definisi Struktur Data (Struct)
3. Fungsi utilitas tampilan (cross platform)
4. Fungsi utama dan menu
5. Logika inti permainan
6. Pengolahan data soal eksternal
7. Sistem penyimpanan progres
8. Sistem papan peringkat

Untuk memberikan pemahaman lebih mendalam mengenai implementasi teknisnya, berikut adalah penjelasan rinci kode program main.c yang diuraikan per blok fungsional:

2.1 Header

```
9   #include <stdio.h>
10  #include <stdlib.h>
11  #include <string.h>
12  #include <conio.h>
13
14  #ifdef _WIN32
15      #include <windows.h>
16  #else
17      #include <unistd.h>
18  #endif
```

Library:

a. stdio.h: digunakan untuk input output dasar. (printf, scanf, fopen).

b. stdlib.h: digunakan untuk konversi tipe data (atoi) dan manajemen memori.

c. string.h: digunakan untuk memanipulasi teks (strcpy, strlen, strtok).

Cross-Platform:

Kode #ifdef _WIN32 digunakan untuk mengecek sistem operasi apa yang digunakan, apabila menggunakan Windows maka menggunakan *library* windows,h sedangkan jika menggunakan linux atau Mac kode akan memanggil library unistd.h. Sehingga program ini dapat dijalankan pada berbagai sistem operasi.

2.2 Definisi Konstanta

```
20  // Warna teks
21  #define GREEN "\033[32m"
22  #define RED "\033[31m"
23  #define YELLOW "\033[33m"
24  #define BLUE "\033[34m"
25  #define CLR "\033[0m"
26
27  // Path folder test (untuk data uji)
28  #define PATH_SOAL "../test/soal.txt"
29  #define PATH_SAVE "../test/save.txt"
30  #define PATH_LEADERBOARD "../test/leaderboard.txt"
31
32  // Durasi delay teks
33  #define DELAY_VFAST 350
34  #define DELAY_FAST 600
35  #define DELAY_MEDIUM 1000
36  #define DELAY_SLOW 1600
37  #define DELAY_VSLOW 2200
```

Mendefinisikan beberapa kode warna (ANSI) yang digunakan untuk mengatur warna teks pada output. Mendefinisikan path atau lokasi file soal, file penyimpanan progres pemain, serta papan peringkat agar seluruh data dapat dimuat dengan benar. Selain itu, mendefinisikan beberapa tingkat durasi delay (dari *very slow* hingga *very fast*), sehingga saat menggunakan fitur delay cukup memanggil fungsi yang telah disediakan.

2.3 Fungsi CLEAR_SCREEN

```
39  void CLEAR_SCREEN() {
40      #ifdef _WIN32
41          system("cls");
42      #else
43          system("clear");
44      #endif
45  }
```

Fungsi ini akan membersihkan layar terminal ketika fungsi dipanggil. Fungsi ini juga akan mengecek sistem operasi sebelum menghapus terminal, karena pada Windows menggunakan kode “cls” sedangkan Linux atau Mac menggunakan koder “clear”.

2.4 Fungsi delay

```
47  void delay(int ms) {
48      #ifdef _WIN32
49          Sleep(ms);
50      #else
51          usleep(ms * 1000);
52      #endif
53  }
```

Fungsi ini akan membuat program memberi jeda (delay) ke eksekusi kode berikutnya. Jeda dalam fungsi ini ditulis dalam milidetik. Jika fungsi ini dipanggil, maka akan memberikan efek “loading” atau memberi waktu pengguna untuk membaca informasi sebelum layar berganti.

2.5 Struct

1. Struct dataPemain

```
55  typedef struct {
56      char nama[100];
57      unsigned int nyawa;
58      unsigned int level;
59      unsigned int skor;
60      unsigned int jumlahBenar;
61  } dataPemain;
```

Struct ini berfungsi sebagai wadah untuk menyimpan semua informasi tentang data pemain, meliputi nama, nyawa tersisa, level, skor, dan jumlah soal benar.

2. Struct Soal

```
63  typedef struct {
64      char pertanyaan[200];
65      char opsi[4][200];
66      int kunci;
67  } Soal;
```

Struct ini berfungsi sebagai wadah untuk menyimpan satu soal, empat pilihan jawaban, dan kunci jawaban yang benar dari pilihan.

3. Struct LeaderboardEntry

```
69   typedef struct {
70       char nama[100];
71       int skor;
72       int level;
73   } LeaderboardEntry;
```

Struct ini berfungsi untuk menyimpan data pemain yang memiliki skor lima teratas. Data yang disimpan dalam struct ini adalah nama, skor, dan level.

2.6 Enum

```
75   enum menu {
76       START = 1,
77       LOAD,
78       LEADERBOARD,
79       CREDITS,
80       EXIT
81   };
```

Dibuat enum dengan nama menu untuk mempermudah dalam kolaborasi pengembangan kode. Dengan menggunakan enum, maka penulisan angka dalam case dapat diganti dengan kata yang lebih mudah dipahami manusia.

2.7 Prototype fungsi

```
83   // Tampilan Judul
84   void banner(void);
85
86   // Sistem Permainan
87   void sistemPermainan(dataPemain *p, Soal *daftarSoal, int totalSoal, int startIndex);
88
89   // Simpan dan Buka File
90   void simpanPermainan(dataPemain *p, int indeksSoal);
91   int muatPermainan(dataPemain *p, int *indeksSoal);
92   void hapusSaveFile();
93   void parseLine(char *line, Soal *s); // Parsing dari file
94   int bukaSoal(Soal *daftarSoal);
95   void updateLeaderboard(char *nama, int skor, int level);
96   void lihatLeaderboard(void);
```

Deklarasi semua fungsi agar program tidak error karena isi fungsi diletakkan setelah fungsi main.

2.8 Fungsi main

```c
int main(void) {
    dataPemain player;
    Soal daftarSoal[20];
    int jumlahSoal = 0;
    int pilihanMenu;

    jumlahSoal = bukaSoal(daftarSoal);
    if (jumlahSoal == 0) return 1;

    while (1) {
        CLEAR_SCREEN();
        banner();

        printf("\n1. MULAI BARU (New Game)\n");
        printf("2. LANJUTKAN (Load Game)\n");
        printf("3. LEADERBOARD\n");
        printf("4. CREDITS\n");
        printf("5. KELUAR\n");
        printf(YELLOW "\nPilih >> " CLR);

        if (scanf("%d", &pilihanMenu) != 1) {
            while (getchar() != '\n');
            continue;
        }
        getchar();

        switch (pilihanMenu) {
            case START:
                // Validasi input nama
                do {
                    printf("Masukkan Nama: ");
                    fgets(player.nama, sizeof(player.nama), stdin);

                    player.nama[strcspn(player.nama, "\n")] = 0;

                    // Cek: Kalau panjang string 0 (Cuma tekan Enter)
                    if (strlen(player.nama) == 0) {
                        printf(RED "Nama tidak boleh kosong!\n\n" CLR);
                    } else if (strlen(player.nama) > 15) {
                        printf(RED "Nama terlalu panjang! Maksimal 15 huruf!\n" CLR);
                        player.nama[0] = '\0';
                    }

                } while (strlen(player.nama) == 0);
```

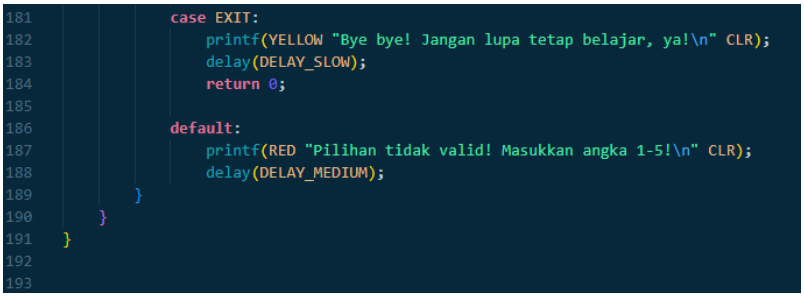
```c
                // Inisialisasi pemain
                player.nyawa = 3;
                player.skor = 0;
                player.level = 0;
                player.jumlahBenar = 0;

                sistemPermainan(&player, daftarSoal, jumlahSoal, 0);
                break;

            case LOAD:
                CLEAR_SCREEN();
                int indeksSoal = 0;
                printf(YELLOW "\nMemuat data pemain...\n" CLR);
                delay(DELAY_FAST);
                 if (muatPermainan(&player, &indeksSoal)) {
                    sistemPermainan(&player, daftarSoal, jumlahSoal, indeksSoal);
                 }
                delay(DELAY_MEDIUM);
                break;

            case LEADERBOARD:
                CLEAR_SCREEN();
                lihatLeaderboard();
                printf("\nTekan 'Enter' untuk kembali...");
                getchar();
                break;

            case CREDITS:
                CLEAR_SCREEN();
                printf(BLUE "\n=== CREDITS ===\n" CLR);
                printf("Code by: Kelompok 1\n");
                printf("Gamma Assyafi Fadhillah Ar Rasyad (L0125013)\n");
                printf("Jaffan Arya Wirasena (L0125017)\n");
                printf("Linda Sihmawati (L0125049)\n");
                printf("\nTekan 'Enter' untuk kembali...");
                getchar();
                break;
```



```
            case EXIT:
                printf(YELLOW "Bye bye! Jangan lupa tetap belajar, ya!\n" CLR);
                delay(DELAY_SLOW);
                return 0;

            default:
                printf(RED "Pilihan tidak valid! Masukkan angka 1-5!\n" CLR);
                delay(DELAY_MEDIUM);
        }
    }
}
```

Fungsi main adalah otak dari seluruh kode ini. Inisialisasi awal program dengan memanggil fungsi bukaSoal untuk memuat data pertanyaan dari file eksternal ke dalam *array* memori. Jika file soal tidak ditemukan atau kosong, program akan langsung berhenti (return 1) untuk mencegah *error* lebih lanjut. Berikutnya program menampilkan berbagai menu yang tersedia, dan meminta input dari pengguna untuk memilih menu. Seluruh logika menu dibungkus dalam perulangan tak terbatas while(1). Hal ini memastikan bahwa setelah pengguna selesai bermain atau melihat skor, aplikasi tidak langsung tertutup, melainkan kembali ke menu utama hingga pengguna secara eksplisit memilih opsi "Keluar" (opsi 5). Saat pengguna memberi masukan input menu, program juga akan memeriksa terlebih dulu apakah yang dimasukkan benar angka atau tidak. Jika pengguna memasukkan karakter huruf, program akan membersihkan *buffer* input (getchar) dan meminta input ulang untuk mencegah *infinite loop* atau *crash*. Kontrol percabangan menu menggunakan switch-case. Program mengarahkan pengguna ke fitur yang sesuai berdasarkan pilihan angka:

i. Case 1 (START): Memulai permainan dari awal sebagai pemain baru. Terdapat validasi input agar nama tidak boleh kosong dan maksimal 15 karakter. Setelah nama valid, status awal pemain di-reset (Nyawa = 3, Skor = 0) dan fungsi sistemPermainan() dipanggil.

ii. Case 2 (LOAD): Melanjutkan permainan berdasarkan save file permainan sebelumnya. Mencoba memanggil fungsi muatPermainan(). Jika file penyimpanan ditemukan dan valid, permainan dilanjutkan dengan status yang tersimpan.

iii. Case 3 (LEADERBOARD): Program akan menampilkan papan peringkat lima pemain teratas yang dapat menyelesaikan permainan dengan skor tertinggi. Program mencoba memanggil fungsi lihatLeaderboard(). Setelah papan peringkat ditampilkan, program menunggu masukan dari pengguna berupa enter untuk kembali ke menu utama.

iv. Case 4 (CREDITS): Program akan mencetak credits dan diawali dengan header berwarna biru. Program menunggu pengguna memberi masukan enter untuk kembali ke menu utama.

v. Case 5 (EXIT): Program menampilkan pesan penutup dan menghentikan program dengan return 0.

2.9 Fungsi Banner

```c
// Menu dan Tampilan
void banner(void) {
    printf(GREEN);
    printf("::::::::: ::::::::: ::::::::: :::    ::: ::::::::  ::::    :::      ::::::::: ::::::::: ::::::::   :::::::  \n");
    printf(":+:    :+: :+:           :+:     :+:   :+: :+:    :+: :+:+:   :+:          :+:  :+:        :+:    :+: :+:   :+: \n");
    printf("+:+    +:+ +:+           +:+     +:+   +:+ +:+    +:+ :+:+:+  +:+         +:+   +:+        +:+    +:+ +:+  :+:+ \n");
    printf("+#++:++#:  +#++:++#      +#+     +#+   +:+ +#++:++#:  +#+ +:+ +#+        +#+    +#++:++#   +#++:++#:  +#+ + +:+ \n");
    printf("+#+    +#+ +#+           +#+     +#+   +#+ +#+    +#+ +#+  +#+#+#       +#+     +#+        +#+    +#+ +#+#  +#+ \n");
    printf("#+#    #+# #+#           #+#     #+#   #+# #+#    #+# #+#   #+#+#      #+#      #+#        #+#    #+# #+#   #+# \n");
    printf("###    ### ##########    ###      ########  ###    ### ###    ####     ######### ########## ###    ###  #######  \n");
    printf(CLR);
}
```

Fungsi ini dipanggil di dalam fungsi main bersamaan dengan menampilkan menu utama, karena fungsi ini dibuat untuk menambah estetika saat berada di menu utama.

2.10 Fungsi sistemPermainan

```c
// Sistem Permainan
void sistemPermainan(dataPemain *p, Soal *daftarSoal, int totalSoal, int startIndex) {
    int i = startIndex;

    for (; i < totalSoal; i++) {
        CLEAR_SCREEN();
        banner();

        unsigned int levelDisplay = 1;
        unsigned int poinSoal = 1;
        char tipeLevel[20] = "DASAR";

        if (i < 5) {
            levelDisplay = 1; poinSoal = 1; strcpy(tipeLevel, "DASAR (1 Poin)");
        } else if (i < 10) {
            levelDisplay = 2; poinSoal = 2; strcpy(tipeLevel, "MENENGAH (2 Poin)");
        } else if (i < 15) {
            levelDisplay = 3; poinSoal = 3; strcpy(tipeLevel, "MAHIR (3 Poin)");
        } else {
            // SYARAT BONUS: Cek p->jumlahBenar
            // Total soal reguler = 15. Syarat > 12 benar.
            if (p->jumlahBenar <= 12) {
                printf(YELLOW "\nJawaban benar kamu (%d), coba lagi untuk dapat level Bonus!\n" CLR, p->jumlahBenar);
                printf("Latihan lagi ya!\n");
                delay(DELAY_SLOW);
                break;
            }
            levelDisplay = 4; poinSoal = 5; strcpy(tipeLevel, "BONUS C++ (5 Poin)");
        }

        printf(BLUE "\nPLAYER: %s | HP: %d | SKOR: %d | BENAR: %d\n" CLR, p->nama, p->nyawa, p->skor, p->jumlahBenar);
        printf(YELLOW "LEVEL %d: %s\n" CLR, levelDisplay, tipeLevel);
        printf("========================================\n");
        printf("SOAL %d: %s\n", i + 1, daftarSoal[i].pertanyaan);
        printf("1. %s\n", daftarSoal[i].opsi[0]);
        printf("2. %s\n", daftarSoal[i].opsi[1]);
        printf("3. %s\n", daftarSoal[i].opsi[2]);
        printf("4. %s\n", daftarSoal[i].opsi[3]);

        printf("\nJawab (1-4) atau ketik 9 untuk SIMPAN & KELUAR: ");
        int jawaban;
```

```
249         if (scanf("%d", &jawaban) != 1) {
250             while(getchar() != '\n');
251             printf(RED "\nHanya angka 1-4 ya! \n" CLR);
252             delay(DELAY_SLOW); i--; continue;
253         }
254 
255         if ((jawaban < 1 || jawaban > 4) && jawaban != 9) {
256             printf(RED "\nPilih angka 1-4 ya!\n" CLR);
257             delay(DELAY_SLOW); i--; continue;
258         }
259 
260         if (jawaban == 9) {
261             p->level = i;
262             simpanPermainan(p, i);
263             return;
264         }
265 
266         if (jawaban == daftarSoal[i].kunci) {
267             printf(GREEN "\nBENAR! (+%d Poin)\n" CLR, poinSoal);
268             p->skor += poinSoal;
269             p->jumlahBenar++;
270             delay(DELAY_MEDIUM);
271         } else {
272             printf(RED "\nSALAH! Jawaban: %d\n" CLR, daftarSoal[i].kunci);
273             p->nyawa--;
274             delay(DELAY_SLOW);
275 
276             if (p->nyawa == 0) {
277                 printf(RED "\nGAME OVER!\n" CLR);
278                 updateLeaderboard(p->nama, p->skor, levelDisplay);
279                 hapusSaveFile();
280                 delay(DELAY_VSLOW);
281                 return;
282             }
283         }
284     }
```

Fungsi ini bertanggung jawab menjalankan seluruh mekanisme *gameplay* kuis interaktif. Di dalamnya terdapat beberapa proses logika utama:

1. Iterasi Soal (*Game Loop*): Menggunakan perulangan for untuk menampilkan soal satu per satu, dimulai dari startIndex (bisa dari 0 untuk permainan baru, atau indeks tertentu jika melanjutkan permainan lalu) hingga total soal habis. Menggunakan perulangan for untuk menampilkan soal satu per satu, dimulai dari startIndex (bisa dari 0 untuk permainan baru, atau indeks tertentu untuk *Load Game*) hingga total soal habis.
2. Sistem Level: Program secara otomatis menentukan tingkat kesulitan dan bobot poin berdasarkan nomor urut soal menggunakan logika if-else: Soal 1-5 (1 Poin), Soal 6-10 (2 Poin). Soal 11-15 (3 Poin), Soal >15 Level Bonus

soal C++ (5 Poin), yang hanya bisa diakses jika pemain menjawab benar minimal 12 soal sebelumnya.

3. Interaksi & Validasi Input: Fungsi ini menampilkan status pemain (Nama, HP, Skor) secara *real-time* di setiap soal. Saat menerima input jawaban, program memvalidasi agar pengguna hanya memasukkan angka 1-4 (untuk menjawab) atau angka 9 (untuk fitur keluar dan simpan).
4. Fitur Simpan & Keluar (*Save & Exit*): Jika pemain memilih angka 9, program akan menyimpan indeks soal terakhir ke dalam variabel p->level, memanggil fungsi simpanPermainan(), dan langsung keluar dari permainan ke menu utama.
5. Mekanisme Jawaban benar dan salah: Jika jawaban benar, maka akan menambahkan skor sesuai bobot level dan menambah penghitung jumlah jawaban benar. Sedangkan jika jawaban salah, nyawa atau HP akan dikurangi. Dan jika nyawa mencapai 0, permainan dihentikan, data skor dicatat ke *Leaderboard*, dan file penyimpanan dihapus (hapusSaveFile()) sehingga pemain harus mengulang dari awal.

2.11 Fungsi simpanPermainan

```
292  void simpanPermainan(dataPemain *p, int indeksSoal) {
293      FILE *file;
294      file = fopen(PATH_SAVE, "w");
295
296      if (file == NULL) {
297          printf(RED "Gagal menyimpan! Pastikan folder 'test' ada!" CLR "\n");
298          return;
299      }
300
301      fprintf(file, "%s#%u#%d#%u#%u#%u",
302              p->nama,
303              p->level,
304              indeksSoal,
305              p->skor,
306              p->nyawa,
307              p->jumlahBenar);
308
309      printf(GREEN "\n[DATA TERSIMPAN] %s berada di Level %u, Soal ke-%d\n"
310          CLR, p->nama, p->level, indeksSoal + 1);
311
312      delay(DELAY_VSLOW);
313
314      fclose(file);
315  }
```

Fungsi simpanPermainan akan dipanggil jika pemain memilih angka 9 saat dalam permainan. Fungsi akan membuka file “save.txt” dengan akses write. Jika ternyata tidak ditemukan file tersebut, maka program akan membuat file dengan

nama yang sama. Data disimpan di dalam file tersebut dengan format pemisah yaitu #. Contoh: KP#2#5#10#3#5. Yang artinya, nama KP, Level 2, Soal indeks ke-5, Skor 10, Nyawa 3, Benar 5.

2.12 Fungsi muatPermainan

```c
int muatPermainan(dataPemain *p, int *indeksSoal) {
    FILE *file = fopen(PATH_SAVE, "r");

    if (file == NULL) {
        printf(RED "Tidak ada data save!\n" CLR);
        delay(DELAY_SLOW);
        return 0;
    }

    int hasil = fscanf(file, "%99[^#]#%u#%d#%u#%u#%u",
            p->nama,
            &p->level,
            indeksSoal,
            &p->skor,
            &p->nyawa,
            &p->jumlahBenar);

    fclose(file);

    if (hasil != 6) {
        printf(RED "Data save rusak!\n" CLR);
        delay(DELAY_SLOW);
        return 0;
    }

    printf(GREEN "\n[LOAD BERHASIL]\n" CLR);
    printf("Player : %s\n", p->nama);
    printf("Level  : %u\n", p->level);
    printf("Soal   : %d\n", *indeksSoal + 1);
    printf("Skor   : %u\n", p->skor);
    printf("Nyawa  : %u\n", p->nyawa);

    delay(DELAY_SLOW);
    return 1;
}
```

Fungsi ini berkebalikan dengan fungsi simpanPermainan. Fungsi ini hanya akan membaca data di file yang sama yaitu “save.txt” tetapi dengan akses r atau hanya baca. Jika ternyata file tersebut tidak ditemukan, maka program akan memberi pesan bahwa tidak ada data yang tersimpan (bisa karena belum pernah save atau permainan terakhir game over). Data yang dibaca juga sesuai dengan format save yaitu dipisahkan dengan #. Setelah dibaca, semua data akan dimasukkan menjadi nilai variabel nama, level, indeksSoal, skor, dan nyawa. Sehingga pemain dapat melanjutkan permainan.

2.13 Fungsi hapusSaveFile

```c
void hapusSaveFile() {
    FILE *file = fopen(PATH_SAVE, "r");
    if (file != NULL) {
        fclose(file);
        remove(PATH_SAVE);
        printf(YELLOW "\n[Save file dihapus karena Game Over]\n" CLR);
    }
    delay(DELAY_SLOW);
}
```

Fungsi ini hanya digunakan ketika pemain mengalami kondisi game over. Seluruh save file akan dihapus dan pemain harus memulai permainan baru jika ingin bermain lagi.

2.14 Fungsi bukaSoal

```
365  int bukaSoal(Soal *daftarSoal) {
366      FILE *file;
367      int jumlahSoal = 0;
368      char buffer[500];
369
370      file = fopen (PATH_SOAL, "r");
371      if (file == NULL) {
372          printf(RED "File soal tidak ditemukan di %s\n" CLR, PATH_SOAL);
373          printf("Cek kembali posisi file .exe kamu ya!\n");
374          return 0;
375      }
376      printf("\nMemuat...");
377      delay(DELAY_MEDIUM);
378      while (fgets(buffer, sizeof(buffer), file)) {
379
380          buffer[strcspn(buffer, "\n")] = 0;
381
382          parseLine(buffer, &daftarSoal[jumlahSoal]);
383          jumlahSoal++;
384      }
385      fclose(file);
386      return jumlahSoal;
387  }
```

Fungsi ini akan membaca data soal di file “soal.txt”. jika ternyata file tersebut tidak ditemukan, maka program akan memberi pesan bahwa soal tidak ditemukan dan mengembalikan nilai 0 untuk mencegah program berjalan tanpa soal. Berikutnya melakukan iterasi pembacaan baris menggunakan fungsi fgets di dalam perulangan while untuk mengambil data per baris dari file teks eksternal dan menyimpannya ke dalam buffer sementara.Sebelum diproses, karakter *newline* (\n) yang secara otomatis terbaca oleh fgets dihapus menggunakan strcspn agar format tampilan teks tidak berantakan. Setiap baris teks mentah dikirim ke fungsi parseLine untuk dipecah menjadi komponen *struct* Soal (pertanyaan, opsi, kunci), sembari menghitung total soal yang berhasil dimuat (jumlahSoal).

2.15 Fungsi parseLine

```
389  void parseLine(char *line, Soal *s) {
390      char *token = strtok(line, "#");
391      strcpy(s->pertanyaan, token);
392
393      for (int i = 0; i < 4; i++) {
394          token = strtok(NULL, "#");
395          strcpy(s->opsi[i], token);
396      }
397
398      token = strtok(NULL, "#");
399      s->kunci = atoi(token);
400  }
```

Fungsi parseLine inilah yang digunakan pada fungsi bukaSoal untuk memecah soal dalam baris teks mentah menjadi *struct* soal yang sudah tersusun rapi. Fungsi ini memotong *string* panjang menjadi bagian-bagian kecil berdasarkan karakter pemisah tanda # dengan menggunakan bantuan fungsi strtok dari *library* string.h. Potongan pertama yang diambil ditandai sebagai teks pertanyaan, kemudian melakukan for loop sebanyak empat kali untuk mengambil potongan-potongan berikutnya sebagai opsi jawaban A, B, C, dan D. Potongan terakhir yaitu kunci jawaban yang awalnya berupa string diubah menjadi integer menggunakan fungsi atoi agar dapat dibandingkan secara numerik dengan input jawaban dari pemain.

2.16 Fungsi updateLeaderboard

```
402  void updateLeaderboard(char *nama, int skor, int level) {
403      LeaderboardEntry entries[100];
404      int count = 0;
405
406      FILE *file = fopen(PATH_LEADERBOARD, "r");
407      char buffer[200];
408
409      if (file != NULL) {
410          while (fgets(buffer, sizeof(buffer), file)) {
411              buffer[strcspn(buffer, "\n")] = 0;
412              char *token = strtok(buffer, "#");
413              if (token != NULL) {
414                  strcpy(entries[count].nama, token);
415
416                  token = strtok(NULL, "#");
417                  if (token) entries[count].skor = atoi(token);
418
419                  token = strtok(NULL, "#"); // Baca Level Lama
420                  if (token) entries[count].level = atoi(token);
421
422                  count++;
423              }
424          }
425          fclose(file);
426      }
427
428      strcpy(entries[count].nama, nama);
429      entries[count].skor = skor;
430      entries[count].level = level; // Simpan level user
431      count++;
432
433      // SORTING (Bubble Sort berdasarkan SKOR tertinggi)
434      for (int i = 0; i < count - 1; i++) {
435          for (int j = 0; j < count - i - 1; j++) {
436              if (entries[j].skor < entries[j + 1].skor) {
437                  LeaderboardEntry temp = entries[j];
438                  entries[j] = entries[j + 1];
439                  entries[j + 1] = temp;
440              }
441          }
442      }
```

```
444      if (count > 5) count = 5; // Top 5
445
446      file = fopen(PATH_LEADERBOARD, "w");
447      if (file == NULL) return;
448
449      for (int i = 0; i < count; i++) {
450          // Format simpan: Nama#Skor#Level
451          fprintf(file, "%s#%d#%d\n", entries[i].nama, entries[i].skor, entries[i].level);
452      }
453      fclose(file);
454  }
```

Fungsi updateLeaderboard bertanggung jawab memelihara daftar juara melalui mekanisme pembaruan otomatis yang dimulai dengan proses pembacaan data lama, yaitu membaca seluruh riwayat data peringkat dari file leaderboard.txt ke dalam *array* memori sementara. Setelah data lama dimuat, fungsi ini menyisipkan profil pemain terbaru ke dalam daftar tersebut dan segera melakukan pengurutan ulang menggunakan algoritma **Bubble Sort** secara *descending* (menurun) berdasarkan perolehan skor tertinggi. Proses ini diakhiri dengan mekanisme limitasi data, di mana program hanya akan menulis ulang lima entri teratas kembali ke file penyimpanan, memastikan efisiensi file eksternal tetap terjaga dan hanya menampilkan pencapaian terbaik.

2.17 Fungsi lihatLeaderboard

```c
void lihatLeaderboard(void) {
    FILE *file = fopen(PATH_LEADERBOARD, "r");
    char buffer[200];
    int rank = 1;

    printf(BLUE "\n================================\n" CLR);
    printf(YELLOW "   HALL OF FAME - RETURN ZER0   \n" CLR);
    printf(BLUE "================================\n" CLR);

    // Header Kolom (Perhatikan spasi dan garisnya)
    printf("No. | %-10s | %-5s | %-3s\n", "Name", "Score", "Lvl");
    printf("---------------------------------\n");

    if (file == NULL) {
        printf(RED "      BELUM ADA JUARA!      \n" CLR);
        printf("---------------------------------\n");
        return;
    }

    // Baca File format: Nama#Skor#Level
    while (fgets(buffer, sizeof(buffer), file)) {
        buffer[strcspn(buffer, "\n")] = 0;

        char *nama = strtok(buffer, "#");
        char *skorStr = strtok(NULL, "#");
        char *levelStr = strtok(NULL, "#"); // Ambil token ketiga (Level)

        if (nama && skorStr && levelStr) {
            printf("%-3d | %-10.10s | %-5s | %-3s\n",
                   rank++, nama, skorStr, levelStr);
        }
    }
    printf("---------------------------------\n");
    fclose(file);
}
```

Fungsi lihatLeaderboard berperan sebagai antarmuka visual untuk menyajikan daftar skor tertinggi kepada pengguna dalam format yang mudah dibaca. Prosesnya diawali dengan mencetak *header* tabel (judul dan nama kolom) agar tampilan terminal terstruktur rapi. Selanjutnya, fungsi ini membuka file “leaderboard.txt” dan membaca data per baris menggunakan fgets. Setiap baris yang dibaca kemudian dipecah menjadi komponen nama, skor, dan level menggunakan strtok, lalu ditampilkan ke layar dengan format perataan kolom yang konsisten. Jika file penyimpanan belum tersedia atau kosong, fungsi ini akan menampilkan pesan bahwa belum ada juara yang terdaftar.

BAB III
PENJELASAN PROGRAM

Setelah memahami *source code*, langkah berikutnya adalah mempelajari bagaimana program kuis ini dioperasikan. Pada bab ini akan dijelaskan tahapan dan opsi yang tersedia saat program dijalankan, mulai dari inisialisasi hingga pengguna memilih menu yang diinginkan.

3.1 Tampilan Awal

Ketika program dijalankan, sistem akan mencoba membaca dan memuat soal dari file soal.txt. Jika file soal tidak ditemukan, program akan menampilkan pesan kesalahan berwarna merah dan berhenti.

Jika file soal ditemukan, program akan menampilkan judul utama “RETURN ZER0” berwarna hijau dan menampilkan beberapa menu utama untuk dipilih oleh pengguna, yaitu:

1. **MULAI BARU (Pilihan 1):** Digunakan untuk mendaftarkan pemain baru dan memulai permainan kuis dari soal pertama (Level 1).
2. **LANJUTKAN (Pilihan 2):** Digunakan untuk memuat dan melanjutkan sesi permainan yang telah disimpan di `save.txt`.
3. **LEADERBOARD (Pilihan 3):** Menampilkan daftar 5 skor tertinggi yang pernah dicapai.
4. **CREDITS (Pilihan 4):** Menampilkan daftar anggota kelompok pembuat program.
5. **KELUAR (Pilihan 5):** Untuk mengakhiri program.

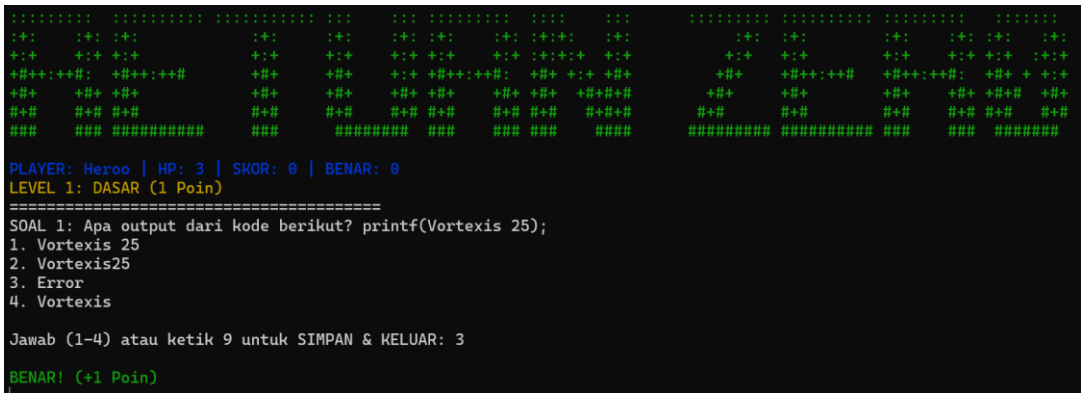
3.2 Tampilan Mulai Baru

Saat pengguna memilih opsi untuk memulai permainan baru, program akan melakukan inisialisasi data pemain dan langsung masuk ke sistem kuis.

1. Pilih opsi 1 pada menu utama
2. Masukkan nama pemain

Setelah inisialisasi data pemain, sistem akan menampilkan status pemain berwarna biru, level soal dan nilai poin yang didapatkan., serta program menampilkan soal dan 4 opsi jawaban.

3. Kemudian, input jawaban.
   a. Jika jawaban benar, program akan menampilkan BENAR! (+ Poin yang didapat). Skor dan jumlah benar akan bertambah.

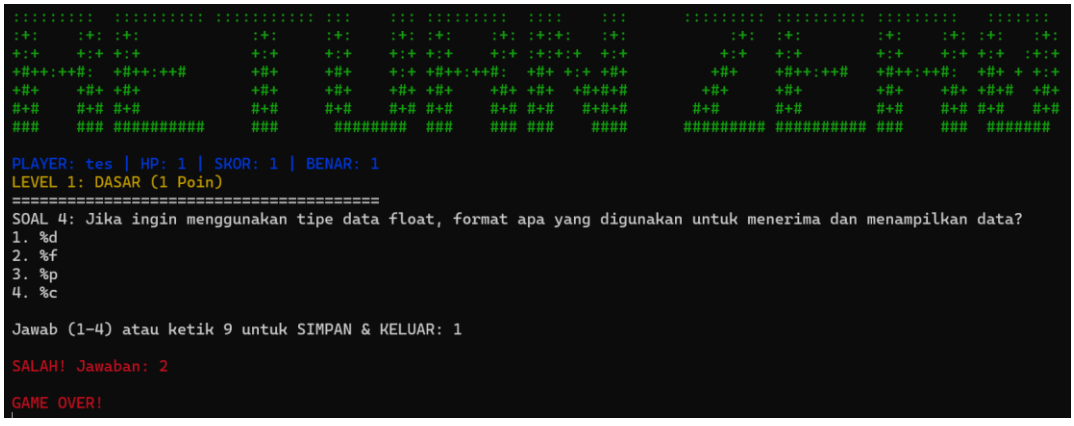


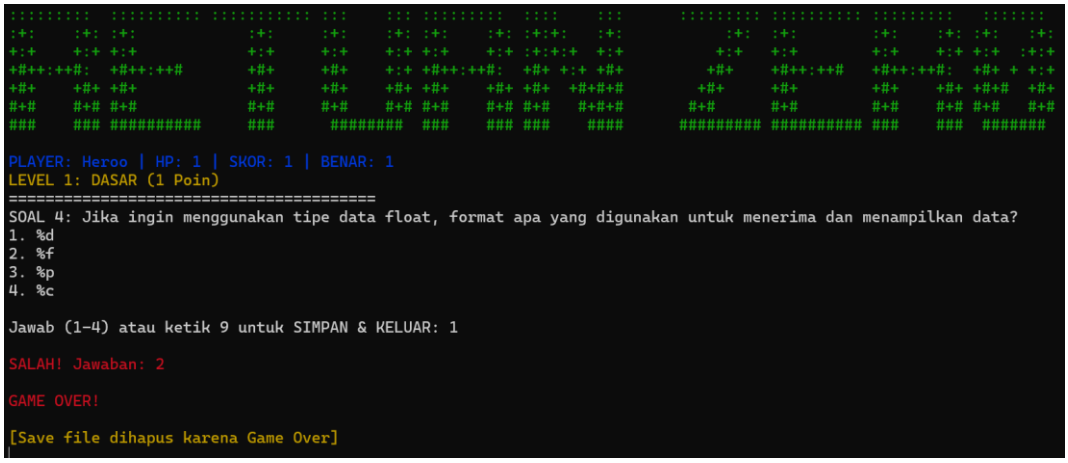
   b. Jika jawaban salah, program akan menampilkan SALAH! Jawaban: [Jawaban yang benar]. Nyawa pemain akan berkurang 1.

c. Jika pemain memilih untuk simpan dan keluar (pilih 9), program akan mencatat data pemain (nama, nyawa, skor, jumlahBenar) dan indeks soal terakhir ke file save.txt. Program akan menampilkan [DATA TERSIMPAN] [Nama] berada di Level [Level], Soal ke-[Nomer Soal]. Program kembali ke Menu Utama.

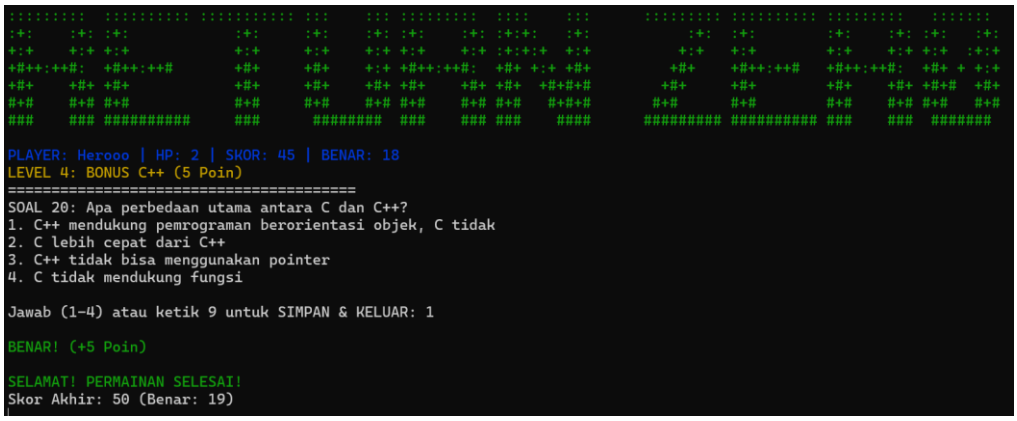
4. Skenario Akhir Permainan

a. Jika Nyawa mencapai 0 setelah menjawab salah, program menampilkan GAME OVER! Dan mencatat skor ke Leaderboard, dan menghapus file save.txt (jika melanjutkan permainan yang sudah disimpan)





b. Jika semua soal berhasil dijawab (dan syarat Level Bonus terpenuhi), program menampilkan SELAMAT! PERMAINAN SELESAI! dan mencatat skor akhir ke *Leaderboard*.

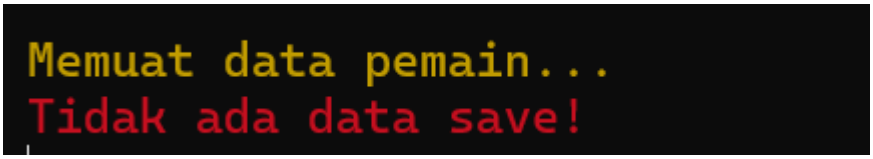


3.3 Tampilan Lanjutkan

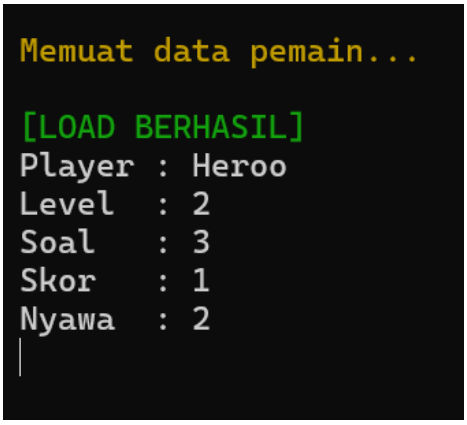
Proses ini digunakan untuk memulihkan sesi permainan yang telah disimpan.

1. Pada menu utama, pilih opsi 2.

2. Pencarian File Save: Program mencoba memuat data dari save.txt.
   a. Jika gagal (file tidak ada atau rusak): Output Tidak ada data save! Program kembali ke Menu Utama.



b. Jika berhasil dimuat: Program menampilkan detail sesi yang tersimpan (Nama, Level, Soal, Skor, Nyawa) dengan pesan [LOAD BERHASIL].



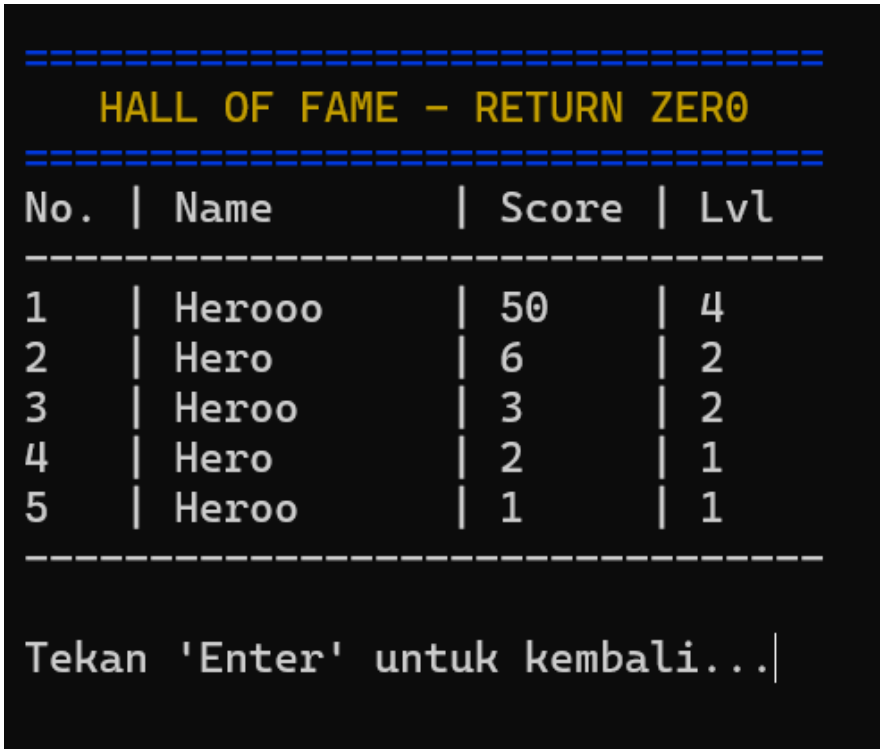
3. Program melanjutkan sistem Permainan dari nomor soal yang tersimpan, dan pemain akan melanjutkan permainan hingga skenario permainan berakhir, seperti di Skenario Akhir Permainan.

3.4 Tampilan Leaderboard

Opsi ini menampilkan daftar 5 pemain dengan skor tertinggi (Hall of Fame).

1. Pada menu utama, pilih opsi 3.

2. Tampilan Daftar: Program menampilkan *banner* HALL OF FAME dan membaca data dari leaderboard.txt. Jika ada data: Menampilkan skor, nama, dan level dalam format tabel.

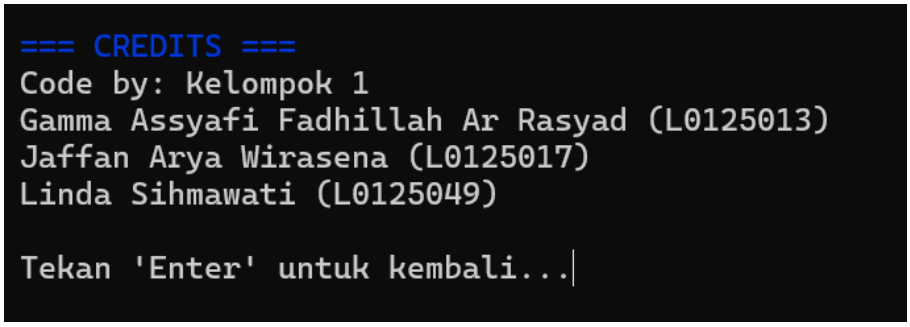


Program menunggu input Enter untuk kembali ke Menu Utama.

3.5 Tampilan Credits

Saat pengguna memilih opsi ke-4, program akan menampilkan daftar anggota kelompok pembuat.

1. Program menampilkan judul **=== CREDITS ===**.
2. Menampilkan Nama dan NIM anggota kelompok.
3. Program menunggu input Enter untuk kembali ke Menu Utama.



3.6 Tampilan Keluar

Saat pengguna memilih opsi ke-5, program akan dihentikan secara total.

1. Program menampilkan pesan perpisahan: Bye bye! Jangan lupa tetap belajar, ya!.
2. Program menghentikan eksekusi.

BAB IV
PEMBAGIAN TUGAS

Pengerjaan proyek ini melibatkan kerja sama tim yang baik. Mulai dari perancangan ide, pembuatan, percobaan program, hingga dokumentasi program. Berikut adalah pembagian tugas tiap anggota:

1. Gamma Assyafi Fadhillah Ar Rasyad (L0125013)
    - Konsep & Perancangan Sistem: Merancang ide dasar permainan, mekanisme gameplay, serta membuat System Design dan Flowchart algoritma.
    - Core Programming: Menulis sebagian besar kode program (main logic) dan mengintegrasikan seluruh modul.
    - Repository Management: Mengelola repositori GitHub, melakukan version control, dan manajemen merge kode.
    - Quality Control (Editor in Chief): Melakukan reviu akhir terhadap penulisan kode dan penyusunan laporan agar sesuai standar.
    - Media & Presentasi: Mendesain presentasi (PPT) (kolaborasi dengan Linda) dan memastikan visualisasi proyek menarik.
    - Manajemen Tim: Mengoordinasikan pembagian kerja dan timeline pengerjaan proyek.

2. Jaffan Arya Wirasena (L0125017)
    - File Processing System: Mengembangkan modul penyimpanan data eksternal untuk fitur Save/Load progres pemain secara terstruktur.
    - Debugging: Melakukan penelusuran bug (error handling) untuk memastikan program berjalan stabil tanpa crash.
    - Dokumentasi Kode (Laporan BAB II): Menyusun penjelasan teknis mengenai source code, struktur data, dan fungsi-fungsi yang digunakan dalam program.

3. Linda Sihmawati (L0125049)
    - Testing: Melakukan uji coba program secara intensif (Black Box Testing) untuk menemukan celah kesalahan logika atau bug.
    - Dokumentasi Pengguna (Laporan BAB III): Menyusun panduan cara menjalankan program (User Manual) agar mudah dipahami pengguna.
    - Project Documentation: Menyusun file README.md untuk repositori dan dokumentasi pendukung lainnya.
    - Media Presentasi: Berkolaborasi dalam penyusunan dan desain Slide Presentasi (PPT).

BAB V
PENUTUP

5.1 Kesimpulan

- Aplikasi "Return Zer0" berhasil diimplementasikan sebagai permainan edukasi berbasis CLI (*Command Line Interface*) yang mampu menguji logika pemrograman C melalui sistem level berjenjang dari Dasar hingga Bonus.
- Penerapan teknik *file handling* terbukti efektif membuat program lebih dinamis karena bank soal dapat diperbarui tanpa kompilasi ulang dan progres pemain bisa disimpan secara persisten.
- Integrasi algoritma *Bubble Sort* pada sistem *leaderboard* dan validasi input yang ketat berhasil menciptakan lingkungan kompetisi yang rapi dan minim *error* saat *runtime*.

5.2 Saran

- Pengembangan antarmuka ke bentuk GUI (*Graphical User Interface*) sangat disarankan agar tampilan visual permainan menjadi lebih menarik dibandingkan format teks konsol saat ini.
- Perluasan basis data soal di dalam file eksternal sebaiknya dilakukan secara berkala untuk mencakup topik pemrograman yang lebih luas dan meningkatkan tantangan permainan.
- Sistem keamanan pada file penyimpanan (`save.txt`) dan *leaderboard* perlu ditingkatkan dengan enkripsi sederhana agar data skor tidak mudah dimanipulasi secara manual oleh pengguna.

5.3 Refleksi

Melalui pengembangan proyek 'Return Zer0', tim kami memperoleh pemahaman komprehensif bahwa pemrograman tidak hanya sebatas penulisan kode, tetapi juga mencakup pengelolaan struktur data yang efisien serta penerapan *file handling* untuk menjaga persistensi data pengguna. Selain memperkuat logika algoritma, khususnya dalam pengurutan data *leaderboard*

menggunakan Bubble Sort, proyek ini memberikan pengalaman nyata mengenai pentingnya kolaborasi tim, *debugging*, dan manajemen repositori dalam menciptakan solusi perangkat lunak yang fungsional, terstruktur, dan terintegrasi dengan baik.

DAFTAR PUSTAKA

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

Deitel, P. J., & Deitel, H. M. (2016). *C How to Program* (8th ed.). Pearson Education.

Gottfried, B. S. (2018). *Schaum's Outline of Programming with C* (4th ed.). McGraw-Hill Education.

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (2nd ed.). Prentice Hall.

Microsoft. (n.d.). *Sleep Function (synchapi.h)*. Microsoft Learn. Retrieved December 16, 2025, from https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-sleep

Prata, S. (2013). *C Primer Plus* (6th ed.). Addison-Wesley Professional.

The Open Group. (2018). *The Open Group Base Specifications Issue 7: unistd.h - Standard Symbolic Constants and Types*. IEEE and The Open Group. https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/unistd.h.html